J-GANNO

ΓΕΝΙΚΕΥΜΕΝΟ ΠΑΚΕΤΟ ΥΛΟΠΟΙΗΣΗΣ

ΤΕΧΝΗΤΩΝ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ

ΣΤΗ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ JAVA

Σύντομη αναφορά στους κύριους στόχους σχεδίασης και τα βασικά χαρακτηριστικά του πακέτου (προέκδοση 0.9Β, Φεβ.1998)

Χάρης Γεωργίου

Αναλυτής-Προγραμματιστής Πληροφοριακών Συστημάτων

email: xgeorgio@compulink.gr

Αθήνα, Μάρτιος 1998

3. ΥΛΟΠΟΙΗΣΗ

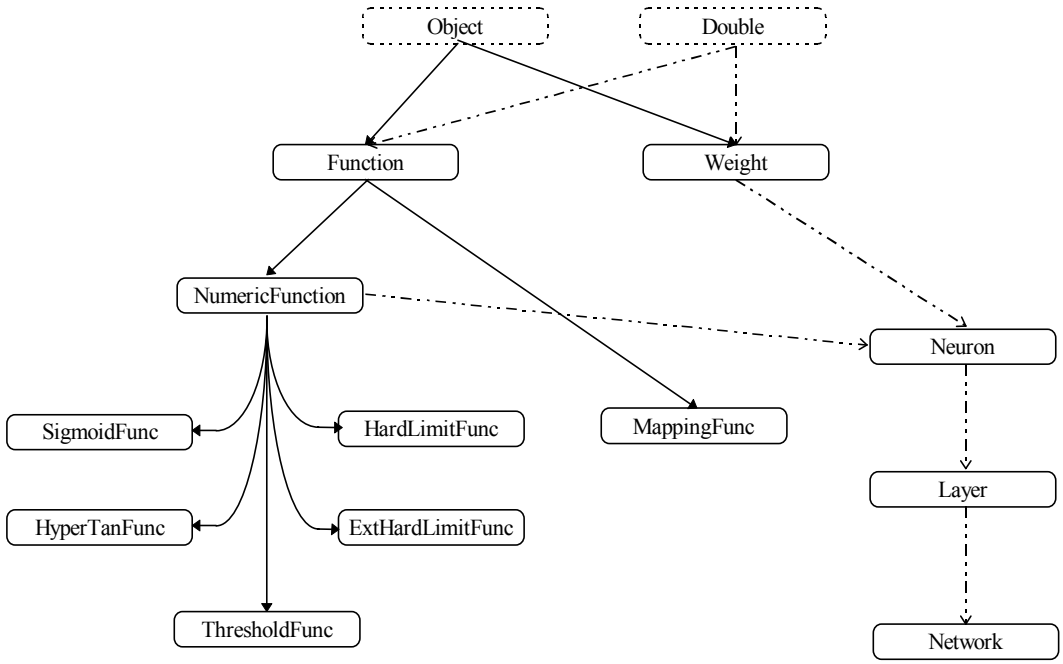
Το πακέτο αποτελείται από ένα πλήθος κλάσεων οι οποίες συνθέτουν το μοντέλο του γενικότερου συστήματος. Επειδή σκοπός είναι η κατασκευή λογισμικού γενικού χαρακτήρα, προσαρμόσιμου ανάλογα με το εκάστοτε πρόβλημα, αλλά χρήσιμου και λειτουργικού όσο αφορά τις εν γένει ενσωματωμένες λειτουργίες, η ανάπτυξη έχει γίνει έτσι ώστε το τελικό αποτέλεσμα να είναι ένα συμπαγές λογισμικό πακέτο, εύκολα επεκτάσιμου και ταυτόχρονα εμπλουτισμένου με πολλές κοινές στα ΤΝΔ διεργασίες (κατασκευή, διασυνδέσεις, ενεργοποίηση, κτλ.).

Εκτός από την ελάχιστη δυνατότητα αυτόματης εκχώρησης και διαχείρισης μνήμης για τις δομές των ΤΝΔ, έχουν υλοποιηθεί γενικευμένες λειτουργίες ενεργοποίησης σε επίπεδο νευρώνων, επιπέδων και δικτύου. Η ενεργοποίηση έχει υλοποιηθεί τόσο με επαναληπτικό (iterative), όσο και με παράλληλο (concurrent) τρόπο εκτέλεσης.

3.1 ΙΕΡΑΡΧΙΚΟ ΜΟΝΤΕΛΟ ΚΛΑΣΕΩΝ

Το παρακάτω διάγραμμα παρουσιάζει σχηματικά με ιεραρχικό τρόπο τη διασύνδεση και τις εξαρτήσεις μεταξύ των κλάσεων του πακέτου. Το στάδιο ανάπτυξης του πακέτου είναι σχετικά πρώιμο, όμως επειδή οι συγκεκριμένες κλάσεις αποτελούν τη βάση ολόκληρου του συστήματος, δεν αναμένονται σημαντικές αλλαγές στο παρακάτω σχήμα.

Οι κλάσεις με διακεκομμένη γραμμή είναι βασικές κλάσεις του Java API που χρησιμοποιούνται ως κύριοι πρόγονοι, ενώ οι υπόλοιπες είναι κατασκευασμένες καθαρά για το συγκεκριμένο πακέτο. Τα διακεκομμένα βέλη δείχνουν την εξάρτηση μέσω μεταβλητών, ενώ τα υπόλοιπα την εξάρτηση σε επίπεδο κληρονομικότητας.



Σχήμα 1: Διάγραμμα Ιεραρχίας Κλάσεων

3.2 ΑΝΑΛΥΤΙΚΗ ΠΕΡΙΓΡΑΦΗ ΚΛΑΣΕΩΝ

Στη συνέχεια περιγράφονται αναλυτικά τα εσωτερικά χαρακτηριστικά των κλάσεων που απαρτίζουν το πακέτο μέχρι αυτή τη φάση της ανάπτυξης. Η περιγραφή γίνεται κατά το δυνατό σε συμφωνία με την ιεραρχία και με την σημαντικότητα στο συνολικό μοντέλο.

Αξίζει να σημειωθεί πως η υλοποίηση των κλάσεων, αν και διευκολύνει την κατασκευή τους με τη χρήση static μεθόδων, κοινών για όλα τα όμοια αντικείμενα, προς το παρόν δεν έχει γίνει κάτι τέτοιο, για να διαφυλαχθεί η δυνατότητα διαχωρισμού τους, αν κάτι τέτοιο κριθεί αναγκαίο στις επόμενες φάσεις της ανάπτυξης.

abstract class Function

Είναι η βασική κλάση-πρόγονος για όλες τις συναρτήσεις που κατασκευάζονται με σκοπό να χρησιμοποιηθούν ως συναρτήσεις μεταφοράς στα νευρωνικά δίκτυα. Στον constructor δίνεται ως παράμετρος String το όνομα-περιγραφή της συνάρτησης. Η κύρια μέθοδος υπολογισμού τιμής (evaluate) δηλώνεται ως abstract και υλοποιείται στους απόγονους ανάλογα με την περίπτωση, ενώ υπάρχει ειδική μέθοδος εκτίμησης (lookup) που χρησιμοποιεί το τελευταίο αποτέλεσμα, αν αυτό συμφωνεί με το ζητούμενο, ώστε να ελαχιστοποιηθεί ο χρόνος υπολογισμού (last-value cache).

abstract class NumericFunction

Είναι αντίστοιχη της προηγούμενης Function, την οποία έχει ως πρόγονο, με εξειδίκευση στις αριθμητικές συναρτήσεις. Αυτό διευκολύνει την κατασκευή και χρήση συναρτήσεων με παραμέτρους και αποτελέσματα τύπου double (native).

class HardLimitFunc

Αποτελεί την πρώτη έτοιμη συνάρτηση μεταφοράς, τη συχνά χρησιμοποιούμενη hard-limit.

$$f(x) = \begin{cases} 1 & , x \geq 0 \\ -1 & , x < 0 \end{cases}$$

Έχει ως πρόγονο την κλάση NumericFunction, την οποία υλοποιεί κατάλληλα, καθορίζοντας ταυτόχρονα και το αντίστοιχο όνομα.

class ExtHardLimitFunc

Αποτελεί επέκταση της προηγούμενης συνάρτησης, όπως συχνά χρησιμοποιείται ως συνάρτηση μεταφοράς.

$$f(x)=\begin{cases} -1 & , x < -\lambda \\ 0 & , -\lambda \le x \le \lambda \\ +1 & , x > \lambda \end{cases}$$

Η παράμετρος (λ) καθορίζεται στον constructor της συνάρτησης, αλλά μπορεί να μεταβληθεί κατά βούληση σε οποιοδήποτε σημείο του προγράμματος.

class ThresholdFunc

Είναι η γνωστή συνάρτηση threshold. Όπως και πριν, η τιμή «κατωφλίου» (θ) καθορίζεται κατά στην αρχικοποίηση του αντικειμένου.

$$f(x)=\begin{cases} 0 & , x < 0 \\ \theta & , \theta \le x \\ x/\theta & , x > \theta \end{cases}$$

class SigmoidFunc

Υλοποιεί τη σιγμοειδή συνάρτηση, με παράμετρο (α).

$$f(x)=\frac{1}{e^{-ax}}$$

class HyperTanFunc

Είναι η συνάρτηση της υπερβολικής εφαπτομένης. Λόγω της μορφής της, δεν υπάρχει καμία παράμετρος εκτός της τιμής εισόδου (x).

$$f(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

Η υλοποίηση έχει γίνει κατά τέτοιο τρόπο, ώστε η εκτίμηση του εκθετικού να γίνεται μόνο μία φορά, ελαχιστοποιώντας το χρόνο υπολογισμού.

class MappingFunc

Η συνάρτηση αυτή στην πραγματικότητα είναι μια γενικευμένη απεικόνιση μεταξύ δύο συνόλων. Επειδή οι τιμές εισόδου και εξόδου είναι γενικά αντικείμενα (Object), ως πρόγονος χρησιμοποιείται η κλάση Function και όχι η NumericFunction όπως οι υπόλοιπες συναρτήσεις μεταφοράς. Κυρίως σκοπεύει στην υλοποίηση μετασχηματισμών εισόδου και εξόδου στα ακραία επίπεδα του νευρωνικού δικτύου, επιτρέποντας έτσι την αυτόματη διαχείριση πραγματικών δεδομένων (raw data). Εσωτερικά

χρησιμοποιεί μια δομή Hashtable, της οποίας το μέγεθος και τα περιεχόμενα μπορούν να οριστούν πριν ή μετά την αρχικοποίηση της συνάρτησης.

$$A \xrightarrow{f(x)} B$$

class Weight

Η κλάση αυτή αποτελεί το πρώτο ουσιαστικό συστατικό στο μοντέλο του framework που υλοποιεί το πακέτο. Στην ουσία, αν και εισάγει το αντικείμενο των «βαρών» του δικτύου, χρησιμοποιείται εκτεταμένα ως γενικός αριθμητικός τύπος κινητής υποδιαστολής. Αυτό γίνεται αφ' ενός για λόγους συμβατότητας μεταξύ των επιμέρους τμημάτων και της εξάρτησης που παρουσιάζουν μεταξύ τους, και εφ' ετέρου ως αποτελεσματική τεχνική επίλυσης του ζητήματος της μεταβλητότητας του τύπου υποδοχής για τα βάρη. Αν και ο τύπος Double που χρησιμοποιείται εν γένει καλύπτει σχεδόν πλήρως τις αριθμητικές απαιτήσεις, μια μελλοντική αλλαγή του βασικού τύπου υποδοχής, πιθανόν για λόγους αριθμητικής ακρίβειας, μπορεί να γίνει ευκολότερα και χωρίς σημαντικές επιπτώσεις στο σύστημα.

Ένα επιπλέον πλεονέκτημα της χρησιμοποίησης ενός γενικού τύπου αριθμητικής κινητής υποδιαστολής είναι το γεγονός ότι προσφέρει σημαντικές διευκολύνσεις στον τομέα του I/O, μια και οι αντίστοιχες δυνατότητες στον τύπο Double είναι οι απολύτως βασικές. Μέθοδοι της κλάσης Weight προσφέρουν τη δυνατότητα διαχείρισης της τιμής της μεταβλητής, την αποτελεσματική διαχείριση της δεσμευμένης μνήμης, κτλ.

class Neuron

Είναι μία από τις τρεις κύριες κλάσεις του μοντέλου. Υλοποιεί με γενικό τρόπο τους νευρώνες, συμπεριλαμβάνοντας όλα τα απαραίτητα χαρακτηριστικά της δομής τους (διάνυσμα εισόδου, διάνυσμα βαρών, συνάρτηση μεταφοράς, μεταβλητές κατωφλίου και εξόδου).

Όλες οι παράμετροι και οι τιμές τους καθορίζονται δυναμικά, δίνοντας έτσι τη δυνατότητα της αποτελεσματικής διαμόρφωσης του κάθε νευρώνα ξεχωριστά κατά την εκτέλεση του προγράμματος. Επίσης δυναμική είναι η δέσμευση και η διαχείριση της μνήμης, ώστε να γίνεται βέλτιστη εκμετάλλευσή της σε όλα τα επίπεδα. Η κατασκευή του διανύσματος βαρών γίνεται εσωτερικά με αυτόματο τρόπο, ανάλογα με την αρχική δομή του αντίστοιχου διανύσματος εισόδου.

Η ενεργοποίηση του νευρώνα είναι η μοναδική «ανώτερη» λειτουργία που μπορεί να υλοποιηθεί στο γενικευμένο μοντέλο. Έτσι, υπάρχουν κατάλληλες μέθοδοι ενεργοποίησης, χρησιμοποιώντας την εκάστοτε δομή και τιμές του διανύσματος εισόδου, του διανύσματος βαρών και των υπόλοιπων παραμέτρων. Η εκτέλεση των αριθμητικών υπολογισμών μπορεί να γίνει στο τρέχον νήμα ελέγχου (thread) του προγράμματος ή σε ξεχωριστό, παράλληλα με άλλες λειτουργίες και με συγκεκριμένη προτεραιότητα.

Για την εύκολη διαμόρφωση του συνολικού δικτύου υπάρχουν ειδικές μέθοδοι για δημιουργία αντίγραφου του συγκεκριμένου νευρώνα, με το ίδιο ή ξεχωριστό διάνυσμα βαρών. Επίσης, υπάρχει η δυνατότητα επέκτασης του διανύσματος βαρών για τη διασύνδεση με κάποιο επιπλέον επίπεδο (layer) του δικτύου, με σκοπό τη δημιουργία ανάδρασης (feed-forward και feedback).

class Layer

Η κλάση αυτή αποτελεί το δεύτερο σημαντικό στοιχείο στην ιεραρχία του μοντέλου. Υλοποιεί γενικευμένα επίπεδα νευρώνων οποιασδήποτε, ουσιαστικά, δομής και συσχέτισης μεταξύ τους. Συμπεριλαμβάνει όλα τα απαραίτητα χαρακτηριστικά για κάθε επίπεδο, όπως κοινό διάνυσμα εισόδου, κοινό διάνυσμα εξόδου και διάνυσμα νευρώνων.

Όπως και πριν, όλες οι παράμετροι και οι τιμές τους καθορίζονται δυναμικά, δίνοντας έτσι τη δυνατότητα της αποτελεσματικής διαμόρφωσης του κάθε επιπέδου ανάλογα με τις ανάγκες του προγράμματος. Επίσης δυναμική είναι η δέσμευση και η διαχείριση της μνήμης, ώστε να γίνεται βέλτιστη εκμετάλλευσή της. Η κατασκευή του διανύσματος των νευρώνων γίνεται εσωτερικά με αυτόματο τρόπο, ανάλογα με τη δομή που υπαγορεύουν το διάνυσμα εισόδου και το διάνυσμα εξόδου του επιπέδου.

Χρησιμοποιώντας την ήδη υλοποιημένη γενικευμένη μέθοδο ενεργοποίησης που έχουν οι νευρώνες, υλοποιούνται αντίστοιχες μέθοδοι ενεργοποίησης του επιπέδου συνολικά. Αξίζει να σημειωθεί πως το σύνολο αυτών των μεθόδων πραγματοποιούν τις δύο διακριτές τεχνικές ενεργοποίησης των νευρώνων, δηλαδή με επαναληπτικό (iterative) ή με παράλληλο (concurrent) τρόπο. Στην πρώτη περίπτωση οι νευρώνες του επιπέδου ενεργοποιούνται διαδοχικά ο ένας μετά τον άλλο, μέχρι να ενεργοποιηθεί ολόκληρο το επίπεδο. Στη δεύτερη περίπτωση ορίζεται ένα «παράθυρο» ενεργοποίησης το οποίο ενεργοποιεί κατά ομάδες τους νευρώνες σε παράλληλα εκτελούμενα νήματα ελέγχου (concurrent threads), το πλήθος και η προτεραιότητα των οποίων καθορίζεται δυναμικά κατά την εκτέλεση του προγράμματος.

Τέλος, για τη δημιουργία συνδέσεων ανάδρασης (feed-forward ή feedback) στο δίκτυο, υπάρχει η δυνατότητα επέκτασης του διανύσματος εισόδου του επιπέδου με το διάνυσμα εξόδου οποιουδήποτε άλλου επιπέδου.

class Network

Αποτελεί την ανώτερη κλάση του μοντέλου στην παρούσα φάση ανάπτυξης. Υλοποιεί γενικευμένα ΤΝΔ, με δυναμικά καθοριζόμενα επίπεδα (layers) και δομή. Περιλαμβάνει, εκτός των βασικών συστατικών (διάνυσμα εισόδου, διάνυσμα εξόδου, διάνυσμα επιπέδων), συναρτήσεις μετασχηματισμών εξόδου και εξόδου, καθώς και πίνακα παραμέτρων.

Η κατασκευή του ΤΝΔ γίνεται και πάλι δυναμικά, δίνοντας μια περιγραφή (με τη μορφή πίνακα ακεραίων) του πλήθους των διαδοχικών επιπέδων και της εσωτερικής δομής του καθενός. Φυσικά κάθε

ειδική διαμόρφωση της αρχικής δομής, όπως για παράδειγμα οι διασυνδέσεις ανάδρασης (feed-forward ή feedback), θα πρέπει να γίνουν αποκλειστικά από το ίδιο το πρόγραμμα.

Τα χαρακτηριστικά και ο τρόπος υλοποίησης της συγκεκριμένης κλάσης ακολουθεί τα πρότυπα των δύο κατώτερων κλάσεων, Neuron και Layer, περιλαμβάνοντας δυναμική διαχείριση μνήμης και δυναμική ενεργοποίηση των νευρώνων του ΤΝΔ. Όσο αφορά στην ενεργοποίηση του δικτύου, αυτή γίνεται φυσικά με διαδοχικό τρόπο για τα επίπεδα (layers) ακολουθώντας τη δομή και τη θέση τους, αλλά η εσωτερική ενεργοποίηση του κάθε επιπέδου μπορεί να γίνει με επαναληπτικό ή παράλληλο τρόπο, χρησιμοποιώντας αντίστοιχες μεθόδους. Με τον τρόπο αυτό το γενικευμένο ΤΝΔ έχει εν γένει τη δυνατότητα ενεργοποίησης των νευρώνων του, με επαναληπτικό ή παράλληλο τρόπο, ανεξάρτητα από τη δομή ή τις εσωτερικές διασυνδέσεις του.

3.3 ΣΧΟΛΙΑ ΚΑΙ ΠΑΡΑΤΗΡΗΣΕΙΣ

Όπως τονίστηκε παραπάνω, βασικό χαρακτηριστικό του μοντέλου αποτελεί ο δυναμικός τρόπος επεξεργασίας σε όλα τα στάδια. Η δυναμική δέσμευση και διαχείριση μνήμης, ο δυναμικός τρόπος κατασκευής της εσωτερικής δομής του ΤΝΔ, καθώς και ο δυναμικός τρόπος ενεργοποίησης των νευρώνων που περιλαμβάνει, προσδίδουν το γενικό χαρακτήρα και την προσαρμοστικότητα του πακέτου συνολικά.

Θα πρέπει, πάντως, να σημειωθούν ορισμένα στοιχεία σχετικά με τη δυναμική επεξεργασία όπως παρουσιάζεται στην υλοποίηση του μοντέλου. Η δυναμική διαχείριση της μνήμης, αν και βελτιστοποιεί την αξιοποίηση του υπολογιστικού περιβάλλοντος, ωστόσο δημιουργεί κάποιες περιπλοκές σε σχέση με τη συγγραφή του κώδικα του προγράμματος. Αυτό οφείλεται κυρίως στο γεγονός ότι οι δυναμικές δομές που χρησιμοποιούνται στη Java (Vector, Hashtable, κτλ.) είναι παράλληλα και πολυμορφικές, κάτι που σημαίνει ότι εκτός από τις διαδικασίες πρόσβασης στα δεδομένα θα πρέπει να εφαρμόζεται αυστηρή τυποθεώρησή τους (casting), σύμφωνα με τις απαιτήσεις της συγκεκριμένης γλώσσας.

Όσο αφορά στη χρήση πολλαπλών νημάτων ελέγχου (threads) για την ενεργοποίηση των νευρώνων, θα πρέπει να τονιστεί πως ενδιαφέρει κυρίως η διαφορά στον τρόπο ενεργοποίησης και λιγότερο οι τελικές επιδόσεις του συστήματος. Σε υπολογιστικά συστήματα με έναν επεξεργαστή, η ενεργοποίηση χρησιμοποιώντας ένα ή περισσότερα threads δεν παρουσιάζει εμφανής διαφορά όσο αφορά στον απαιτούμενο χρόνο επεξεργασίας, ωστόσο η διαδικασία ενεργοποίησης των νευρώνων διαφέρει κατά πολύ. Φυσικά, ο τελικός σκοπός αυτής της διαφοροποίησης, όπως και στο ζήτημα της (δυναμικής) διαχείρισης της μνήμης, είναι πάντα η βέλτιστη αξιοποίηση των πόρων του συστήματος.